Application Note
January 2002

Supplemental Logic and Interconnect Cell (SLIC) *ORCA*® Series 3 FPGAs

Introduction

This application note features the *ORCA* Series 3 Supplemental Logic and Interconnect Cell (SLIC). This cell provides in each PLC high-performance, 3-statable bidirectional buffers, fast decode logic, and other flexible combinations of buffer and decode logic. The SLIC can be used for fast decode, wide fan-in logic functions, and quick 3-state buffering. In this note, SLIC benefits with regard to speed, area and routability, and flexibility will be described. The SLIC architecture will then be detailed with these benefits in mind. The note also includes how the SLIC may be used, how to instantiate SLIC elements, and specific applications and examples using the SLIC.

Supplemental Logic and Interconnect Cell Benefits

The SLIC offers many performance and efficiency benefits described as follows.

SLIC Benefits

- **Flexibility:**
  - — Allows for several modes of operation (internal 3-state buffers, wide fan-in logic and decoder) that can be mixed depending upon the application.
  - — Provides flexible logic functions such as decode and 3-statable output drivers in the same SLIC.
  - — Allows for a standard interface to RAMs; a single data bus can be used, even for large RAM structures.
- **Speed:**
  - — Fast decode functions provide up to ten inputs in one level of logic with no routing delays, including optional input inversion per input.
  - — Wide fan-in functions that use fewer logic levels than LUTs (provides up to ten INPUT AND/OR/AOI functions) depending upon the fan-in.
  - — SLICs have fast connectivity to input buffers, thus providing fast edge decoder capability.
  - — 3-state buffers:
    - — Provide up to eight 3-state drivers within each PLC (grouped into two nibbles).
    - — Independent 3-state control of each nibble.
    - — Provide ability for fast multiplexing of general signals and data buses.
    - — Allow for true bidirectional buses.
    - — Provide fast and efficient multiplexing of address and data buses used in RAMs.
- **Area and Routability:**
  - — Frees up LUTs for other more general functions.
  - — Direct connections from PFU to SLIC (see Figure 2) allow logic to drive fast internal 3-state buses.
  - — Reduces required routing for data path applications.
  - — SLIC buffers are used to provide routing to diagonal PFUs.
  - — SLICs also used as routing repowering buffers to increase system speed.

Architecture/Functionality

In this section, the architecture will be described in detail as it relates to SLIC benefits of flexibility, speed, density, and routability.

Overview

Each PLC contains a SLIC that resides outside of the PFU as shown in Figure 1 below. In general, each SLIC consists of three primary groups which may be configured in logic (decoder) or buffer (3-state bidirectional) modes.

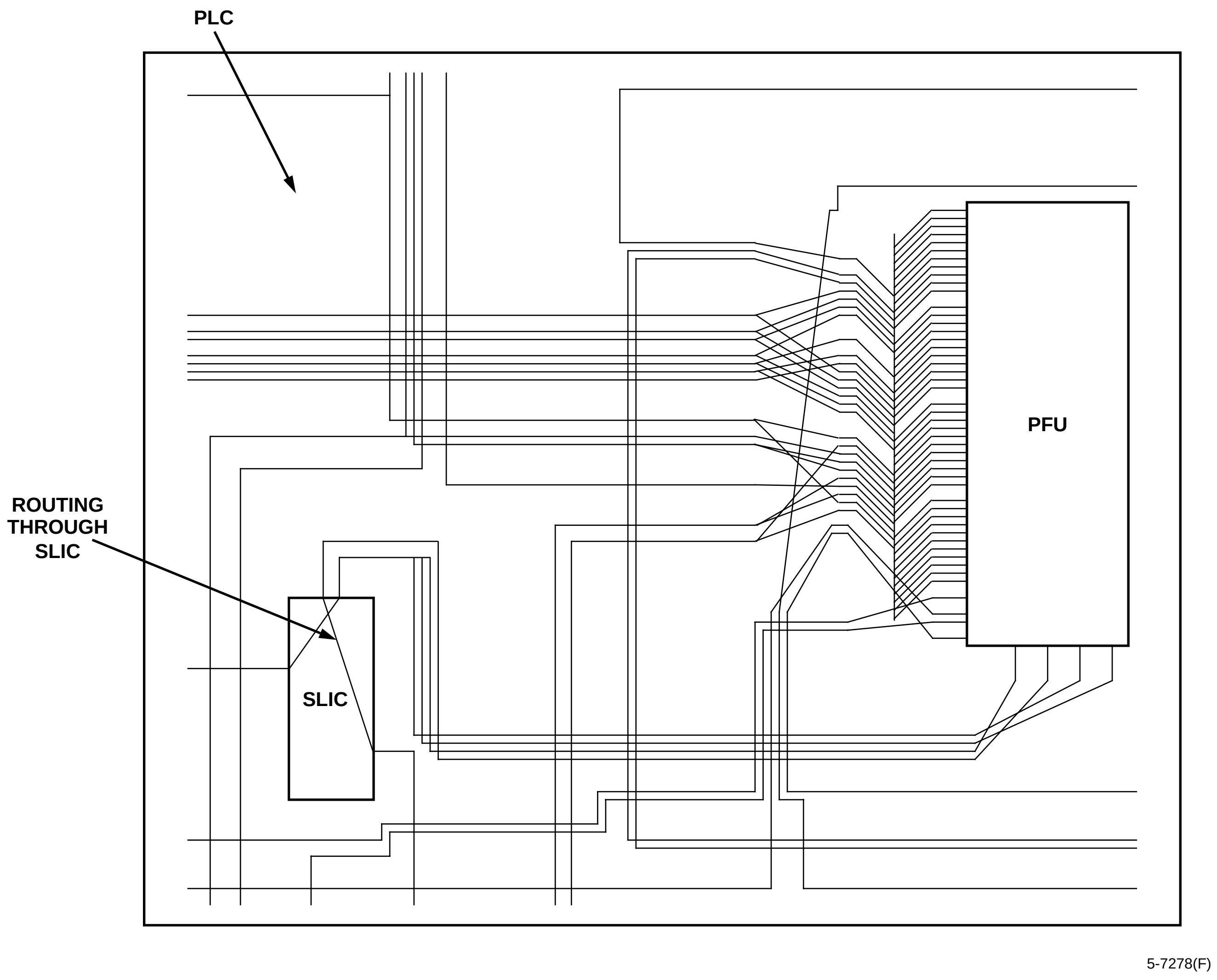


Figure 1. PLC with PFU and SLIC

Flexibility

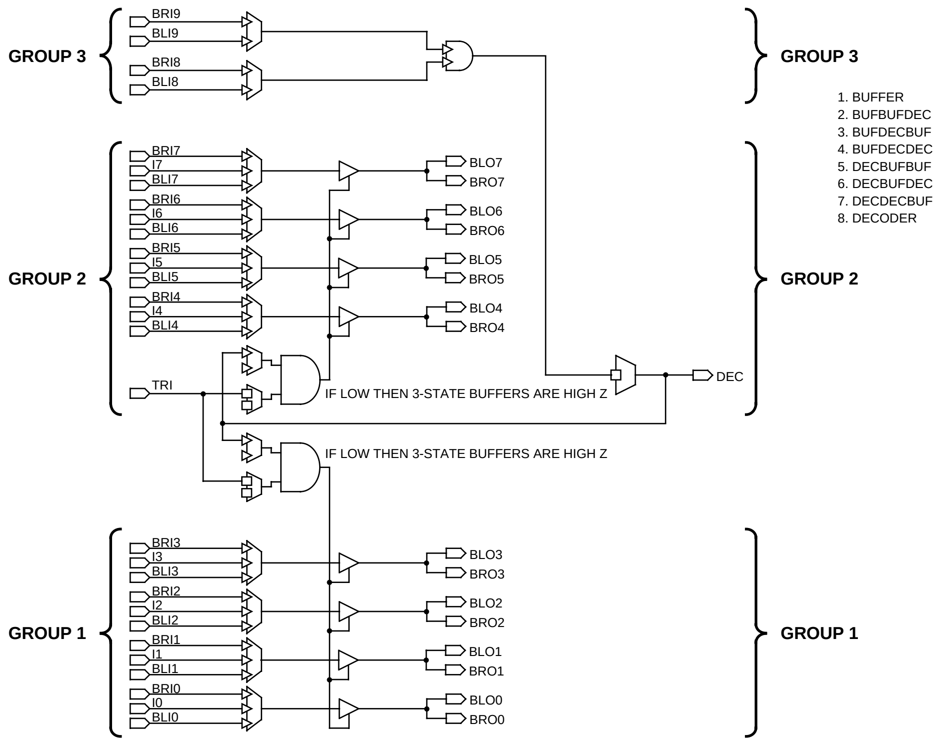
The SLIC architecture provides flexibility in its logic to generate decoder or other combinatorial logic functions. It can be configured to operate in several different modes. The SLIC is partitioned into three primary groups: two groups of 4 bits, and one group of 2 bits for a total of up to ten inputs. Each individual grouping may be configured into either a decoder (AND gate) or buffer (bidirectional 3-state buffer) mode independently.

Architecture/Functionality (continued)

Flexibility (continued)

General Architecture

The general SLIC architecture, as depicted in EPIC, is illustrated in Figure 2. This figure illustrates the BUFBUF-DEC mode. Each buffer group can be input to an AND gate (2-input AND for Group three, 4-input for Groups one and two). This is shown for Group 3 in Figure 2. When used in this mode, any of the inputs to the AND gates may be individually inverted or tied to a logic 1 (VHI). The AND gate outputs can then be connected to a 3-input gate that may be configured as an AND gate or an OR gate. This capability will be shown in more detail later in this note. The output of the 3-input gate may also be inverted (to create an AOI, for example).



5-7279(F)

Figure 2. SLIC Architecture

The two groups of four buffers have 3-state capability while the top group of two buffers does not. There is one 3-state control input (TRI) in each SLIC. The decoder output (DEC) from the same SLIC can also be used for 3-state control. For each group of four bidirectional buffers, the TRI signal may also be inverted, disabled, or ANDed with the decoder output (DEC) signal (driven by any groups not being used as 3-state buffers).

As described previously, each grouping may be configured into particular modes independently. The SLIC modes are described in the next section. It should also be noted that a fast connection is provided from the DEC output to the ninth PFU register to allow fast synchronous operation through the SLIC.

Architecture/Functionality (continued)

Flexibility (continued)

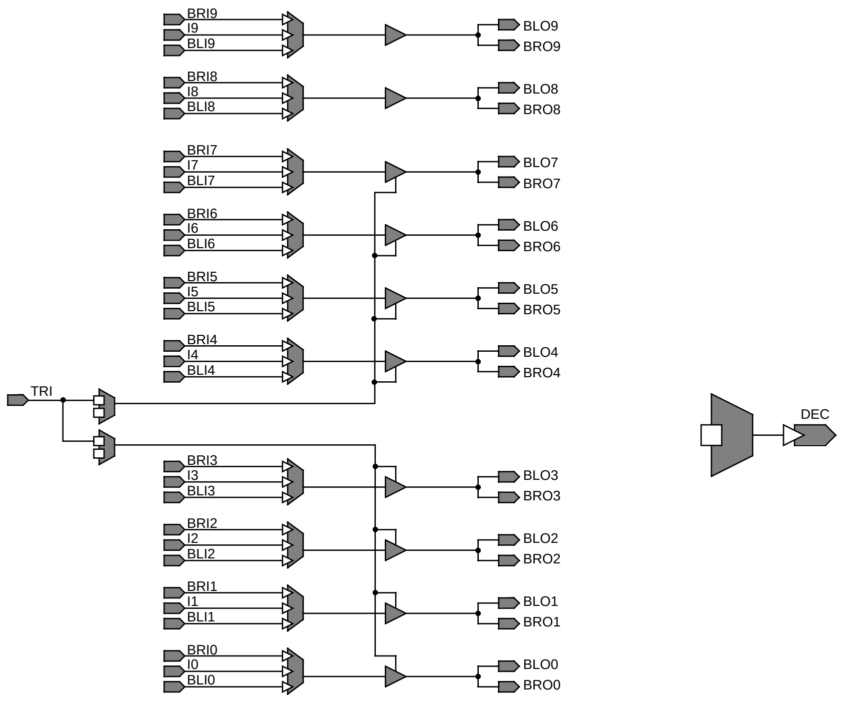
SLIC Modes

The flexibility of the SLIC allows it to operate in eight different modes. Table 1 describes the modes. Each of the groups described in the previous section can operate as an independent function depending upon the mode. The mode names can best be described as Group 1, Group 2, Group 3.

Table 1. SLIC Modes of Operation

Mode #	Mode	BUF[3:0] Group 1	BUF[7:4] Group 2	BUF[9:8] Group 3
1	BUFFER	Buffer	Buffer	Buffer
2	BUFBUFDEC	Buffer	Buffer	Decoder
3	BUFDECBUF	Buffer	Decoder	Buffer
4	BUFDECDEC	Decoder	Decoder	Decoder
5	DECBUFBUF	Decoder	Buffer	Buffer
6	DECBUFDEC	Decoder	Buffer	Decoder
7	DECDECBUF	Decoder	Decoder	Buffer
8	DECODER	Decoder	Decoder	Decoder

BUFFER Mode. In Figure 3, BUFFER Mode #1 is shown. This mode consists of ten bidirectional, twin-quad buffers. Eight are 3-state buffers and the top grouping of two are regular buffers. The TRI signal controls the enables for the two sets of nibblewide 3-state buffers. The TRI signal can also be set to be ignored independently for each group, thus providing a group of regular buffers that can be used to repower general routing signals. In this mode, the DEC output may be used to generate a constant 1 (VHI) or 0 (VLO) for general use.



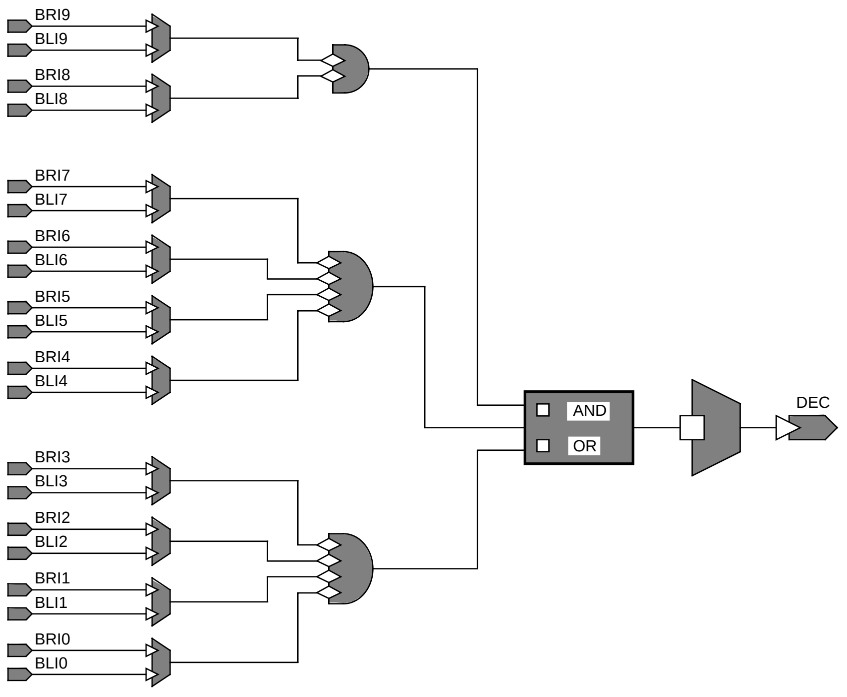
5-7281(F)

Figure 3. BUFFER Mode #1

Architecture/Functionality (continued)

Flexibility (continued)

DECODER Mode. In Figure 4, DECODER mode #8 is illustrated. This mode allows for up to ten inputs of logic. Any combination of inputs may use the AND/NAND function, AND-OR, or AND-OR-INVERT. Flexibility is provided to enable any input or output to be inverted. This also allows logic such as a ten-input OR/NOR to be created.



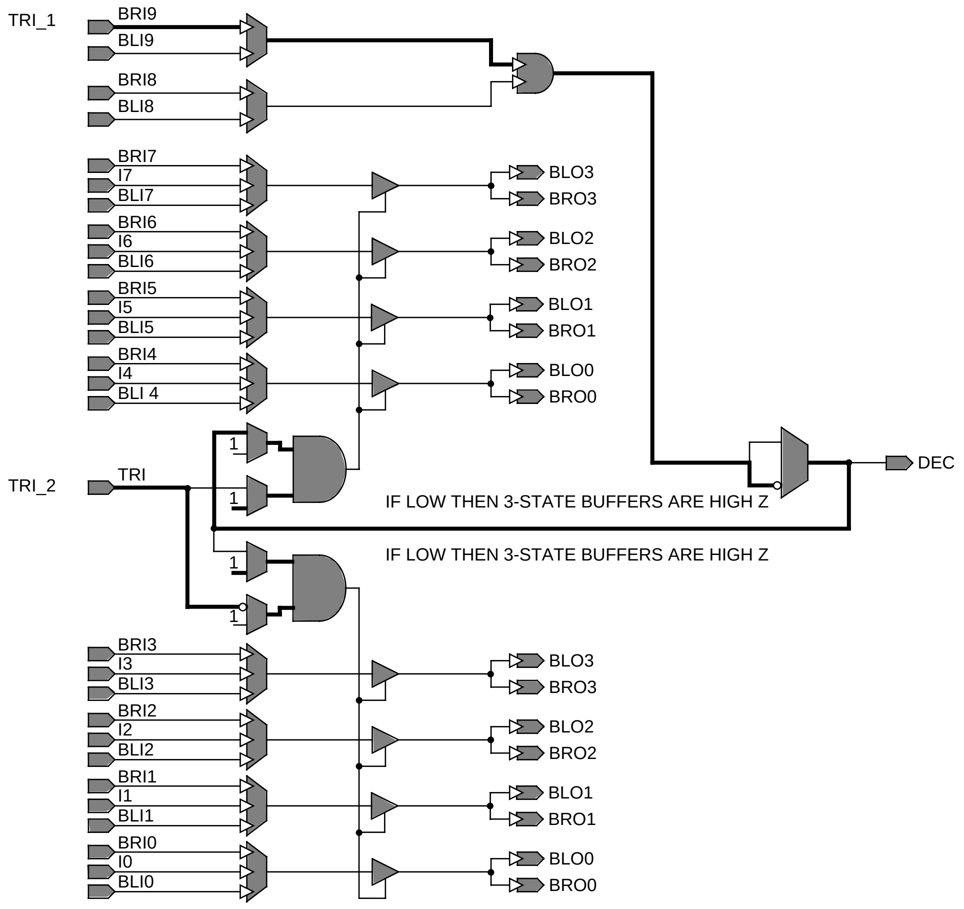
5-7282(F)

Figure 4. DECODER Mode #8

Architecture/Functionality (continued)

Flexibility (continued)

BUFBUFDEC Mode. BUFBUFDEC mode #2, is depicted in Figure 5. It illustrates the SLIC in a mixed mode. The bottom two groups are configured in BUFFER mode and the top group is in DECODER mode. This example illustrates how the SLIC may be used for two truly independent nibblewide 3-state buses with separate 3-state controls. The TRI_2 input is connected to the TRI input of the SLIC and used to control Group 3. TRI_1 is connected to the Group 1 decoder AND gate and fed back to the Group 2, 3-state control via the DEC output.



5-7283(F)

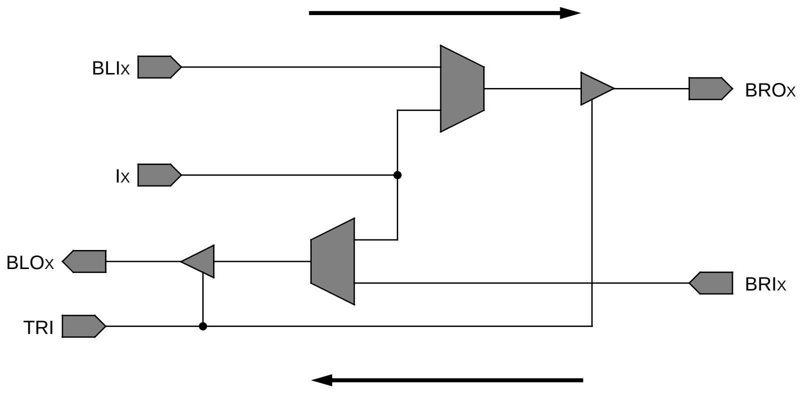
Figure 5. BUFBUFDEC Mode #2

Architecture/Functionality (continued)

Flexibility (continued)

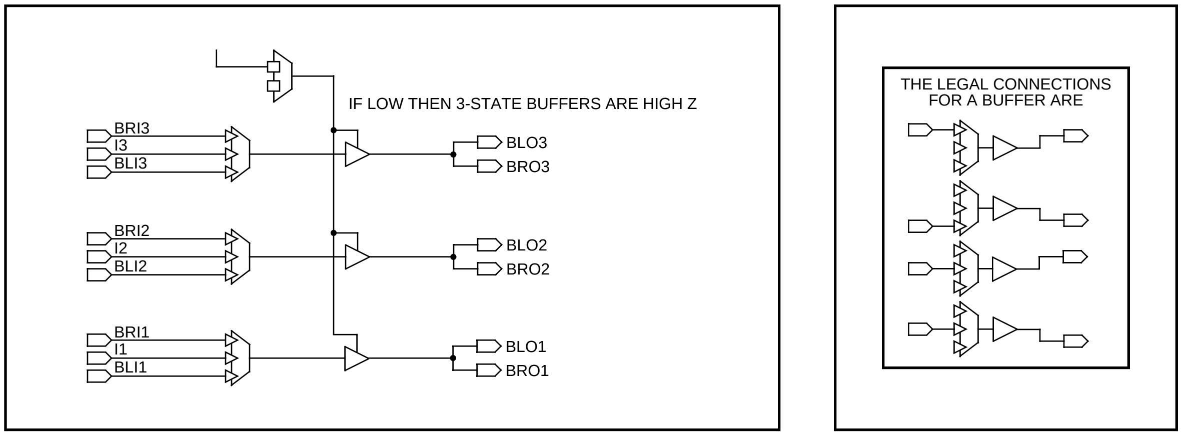
Buffers

In the BUFFER mode or grouping, the SLIC consists of up to ten bidirectional buffers. As depicted in EPIC, each buffer has three choices of input and two choices of output. Each buffer can drive from the left to the right, the right to the left, or from the central input to either the left or right output, or both. The central input comes directly from the PFU outputs (O[9:0]). These are used for high-speed connections between the PFU and SLIC. Figure 6 illustrates the BIDI buffers in schematic form, and Figure 7 shows the buffers as depicted in EPIC. Since the EPIC representation looks quite different, the EPIC drawing has a key to show the legal combinations of inputs and outputs. A right input (BRIx) must go out on a left output (BLOx), a left input (BLIx) must go out on a right output (BROx) and a central input (Ix) can go out on either the right or left outputs or both. If EPIC is not used, choices of inputs and outputs are generally done automatically via the map, place and route tools in *ORCA* Foundry.



5-7284(F)

Figure 6. Bidirectional Buffers


IF LOW THEN 3-STATE BUFFERS ARE HIGH Z
BRI3
I3
BLI3
BLO3
BRO3
BRI2
I2
BLI2
BLO2
BRO2
BRI1
I1
BLI1
BLO1
BRO1
THE LEGAL CONNECTIONS FOR A BUFFER ARE

5-7285(F)

Figure 7. 3-State Buffers in EPIC and EPIC Key

Architecture/Functionality (continued)

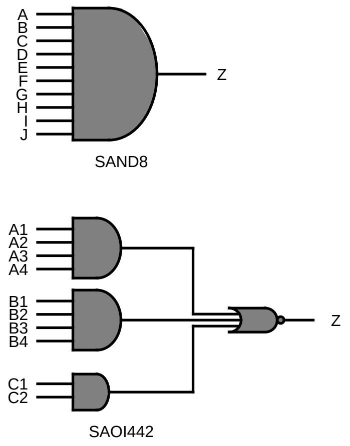
Flexibility (continued)

SLIC Library Cells

There are several library cells in the *ORCA* Series 3 library that may be used to map to a SLIC. These cells are listed below and some examples are shown in Figure 8.

SLIC AND CELLS	SLIC AND-OR-INVERT (AOI) CELLS
SAND2	SAOI42
SAND4	SAOI44
SAND6	SAOI442
SAND8	
SAND10	

SLIC OR CELLS
SOR2
SOR4
SOR6
SOR8
SOR10



5-7286(F)

Figure 8. SAND8 and SAOI442

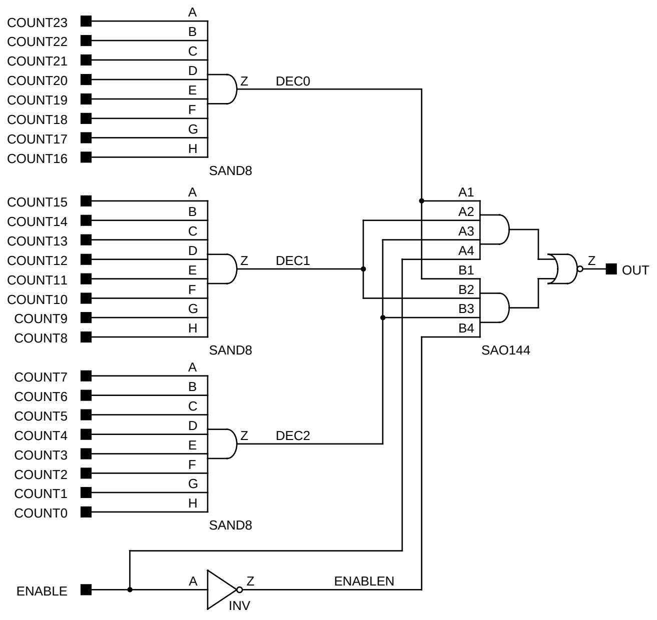
Architecture/Functionality (continued)

Flexibility (continued)

Instantiating a SLIC Decoder

The following example in Figure 9 illustrates how to instantiate the SLIC decoder using VHDL. To invert any of the inputs or outputs of any of the SLIC library cells, instantiate inverters where needed and the mapper in *ORCA* Foundry will pass the inversion to the SLIC cell.

```
SLIC1: SAND8 port map(A=>COUNT23,
                      B=>COUNT22,
                      C=>COUNT21,
                      D=>COUNT20,
                      E=>COUNT19,
                      F=>COUNT18,
                      G=>(COUNT17,
                      H=>COUNT16,
                      Z=>DEC0);
SLIC2: SAND8 port map (A=>COUNT15,
                      B=>COUNT14,
                      C=>COUNT13,
                      D=>COUNT12,
                      E=>COUNT11,
                      F=>COUNT10,
                      G=>(COUNT9,
                      H=>COUNT8,
                      Z=>DEC1);
SLIC3: SAND8 port map (A=>COUNT7,
                      B=>COUNT6,
                      C=>COUNT5,
                      D=>COUNT4,
                      E=>COUNT3,
                      F=>COUNT2,
                      G=>(COUNT1,
                      H=>COUNT0,
                      Z=>DEC3);
SLIC4: SAOI44 port map (A1=>DEC0,
                      A2=>DEC1,
                      A3=>DEC2,
                      A4=>ENABLEN,
                      B1=>DEC),
                      B2=>DEC1
                      B3=>DEC2
                      B4=>ENABLE,
                      Z=>OUT);
EN_INV: INV port map (A=>ENABLE, Z=>ENABLEN);
```



5-7287(F)

Figure 9. VHDL SLIC Instantiation

Speed

The SLIC provides greater performance through its fast decode functions. Decoding of up to ten inputs is provided in one level of logic with no routing delays. The following compares the performance of using a SLIC as an 8-bit decoder vs. a two-level softwired look-up table (SWL) in the Series 3 *ORCA*. If the SWL is not used, delays will be increased further due to routing.

Table 2. SLIC Decoding

8-Bit Decoder	3Txx-5	3Txx-6
SLIC	1.80 ns	1.36 ns
LUT	3.6 ns	2.8 ns

Architecture/Functionality (continued)

Speed (continued)

Wide Logic

The SLIC is very useful for many wide logic applications. In general, a wide decode will be faster with a SLIC decoder implementation than a LUT implementation. However, there are instances where an LUT decode will be as fast or faster than a SLIC decode. Depending upon the fan-in of the design, if the number of SLIC levels exceed the number of LUT levels required to implement the same design, then the LUT implementation would be faster. Figures 10 and 11 show the number of SLIC levels required vs. fan-in. The gray areas show the number of inputs in which there will be fewer levels of SLIC logic. Thus, instantiating SLIC elements will yield faster results. Fan-in in the white areas will use the same number of logic levels with either SLIC elements or LUTs. Performance will depend upon routing delays between levels. If a softwired LUT is used, then it may be faster than two levels of SLIC logic.

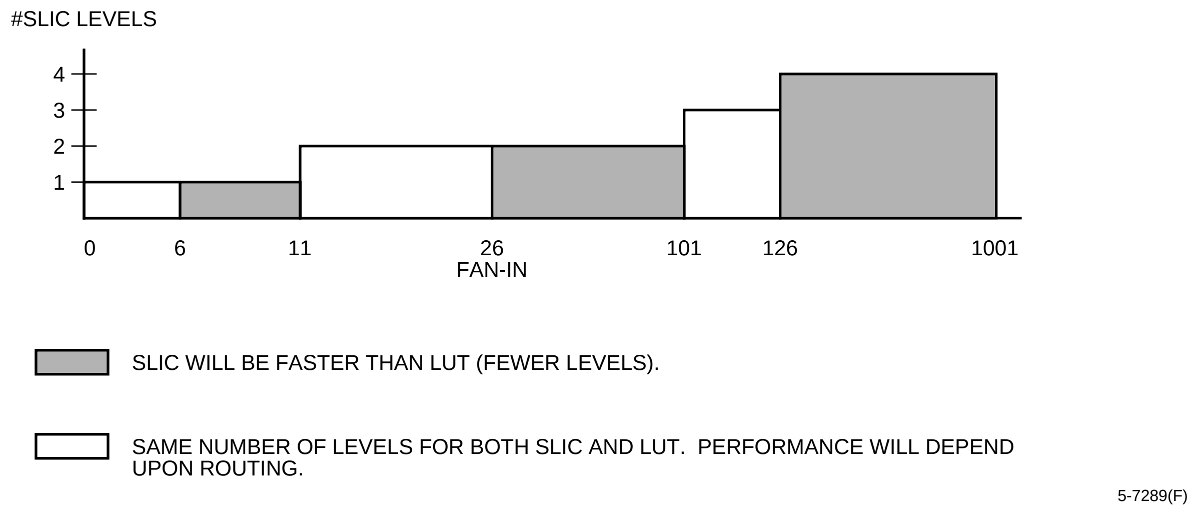


Figure 10. SLIC Levels

Architecture/Functionality (continued)

Speed (continued)

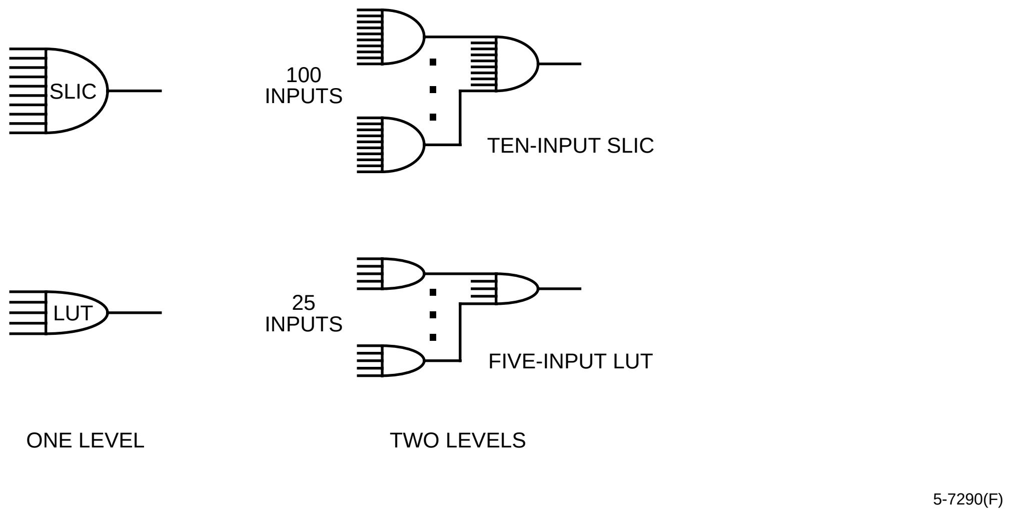


Figure 11. SLIC vs. LUT, One and Two Levels of Logic

PIC Routing

The programmable input/output cells (PIC) are located along the perimeter of the device. The PIC routing includes a fast path from the input pins to the SLICs in each of the three adjacent PLCs (one orthogonal and two diagonal). This allows for input signals to be quickly processed by the SLIC decoder, and gives it the capability to act like a fast I/O signal decoder (sometimes called an edge decoder).

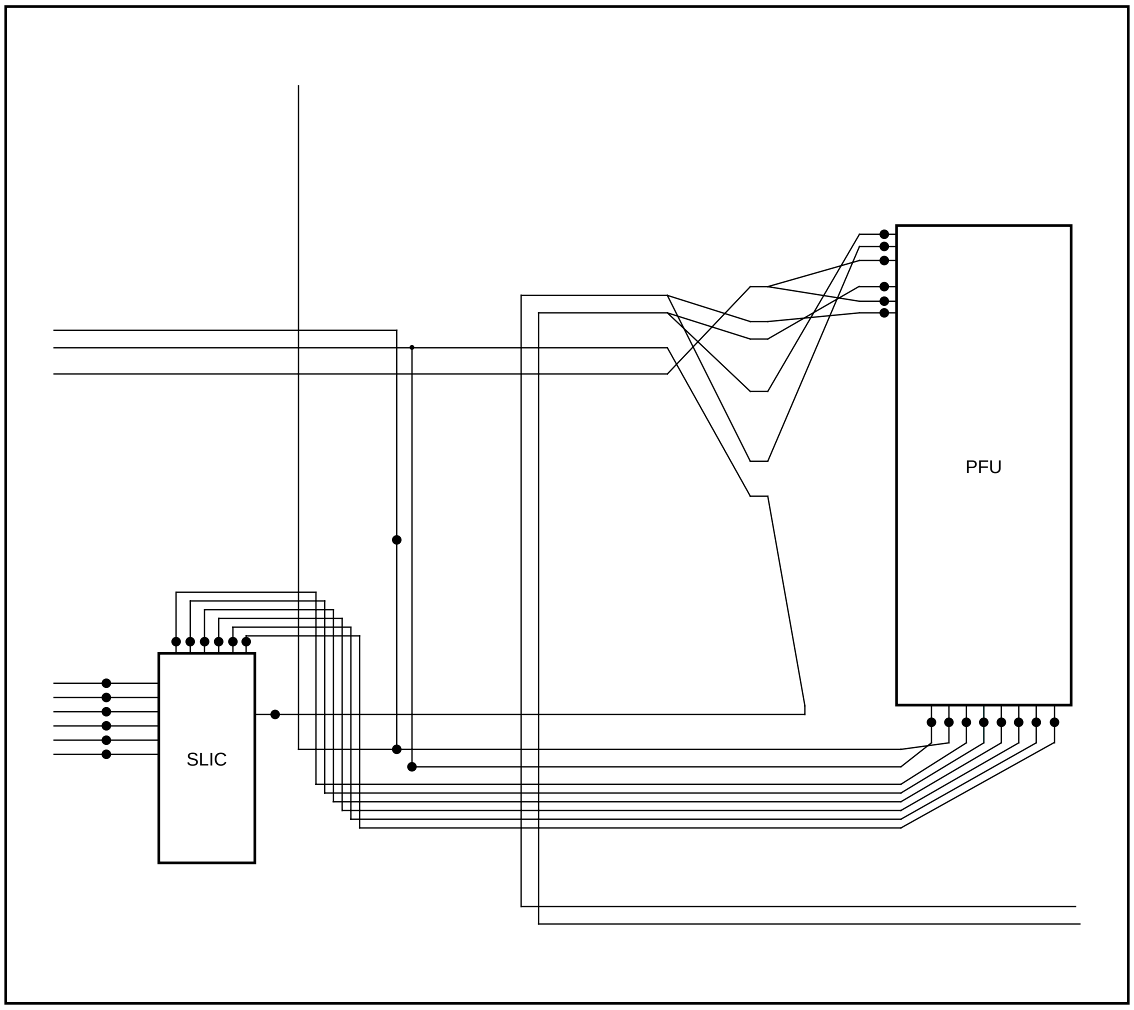
Architecture/Functionality (continued)

Area/Routability

In addition to speed and flexibility, the SLIC provides density and routability advantages. By taking advantage of SLIC logic, LUTs may be used and optimized for other functions, thus reducing area that would have been required by using only LUT logic.

Figure 12 shows the direct connections between the PFU and SLIC devices. These are very fast connections to enhance routability. They also provide very fast internal 3-state buses. This SLIC alignment is done automatically by the mapper and allows the place and route tool to use the direct connections, thus minimizing routing between the PFU and SLIC devices.

Note that when creating hard macros with SLICs, PFU and SLIC ports may not be aligned since the macro containing the SLIC will appear to be a black box to the mapper. It is therefore recommended that for any hard macro with a SLIC, the data source PFU with the proper alignment be included to facilitate routability.



5-7291(F)

Figure 12. Direct Connections Between the PFU and SLIC

Applications

Modulo N Counter

The SLIC logic provides a quick and convenient method for creating Modulo N counters. The SLIC is used for decoding the counter outputs. The result is used to reset the counter. Figure 13 shows an example of a Mod 10 or BCD up-counter by instantiating the SAND2 component. If the counter is set to a specific value, M during reset, then the counter becomes a modulo N-M counter.

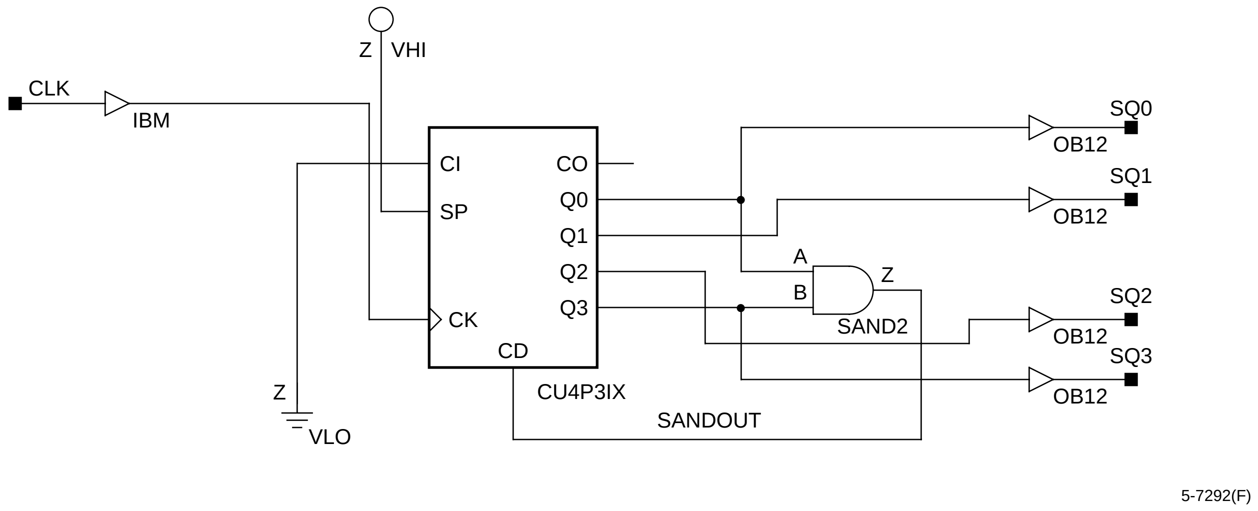


Figure 13. Modulo 10 Counter

Applications (continued)

RAM Implementation

The SLIC is also used for address decoding and 3-state data out buffering in RAM generation. Figure 14 illustrates a 128 x 8 RAM. Each Series 3 PFU can be used to generate a 32 x 4 synchronous, dual-port RAM. Wider memories can be created by using two or more PFU RAMs in parallel, with the same address and control signals, but with different nibbles of data. To achieve depth expansion, write address bits are used for the write port enables. In the example, WADDR bits 5 and 6 are decoded (via a SLIC decoder) to enable the corresponding banks of RAM. The read address bits are also decoded similarly to control the enables of the 3-state buffers which allow read data to share the same data bus as the write data bus, if needed. In this way, high-performance RAMs may be generated quickly and efficiently without using additional PFUs for decode logic.

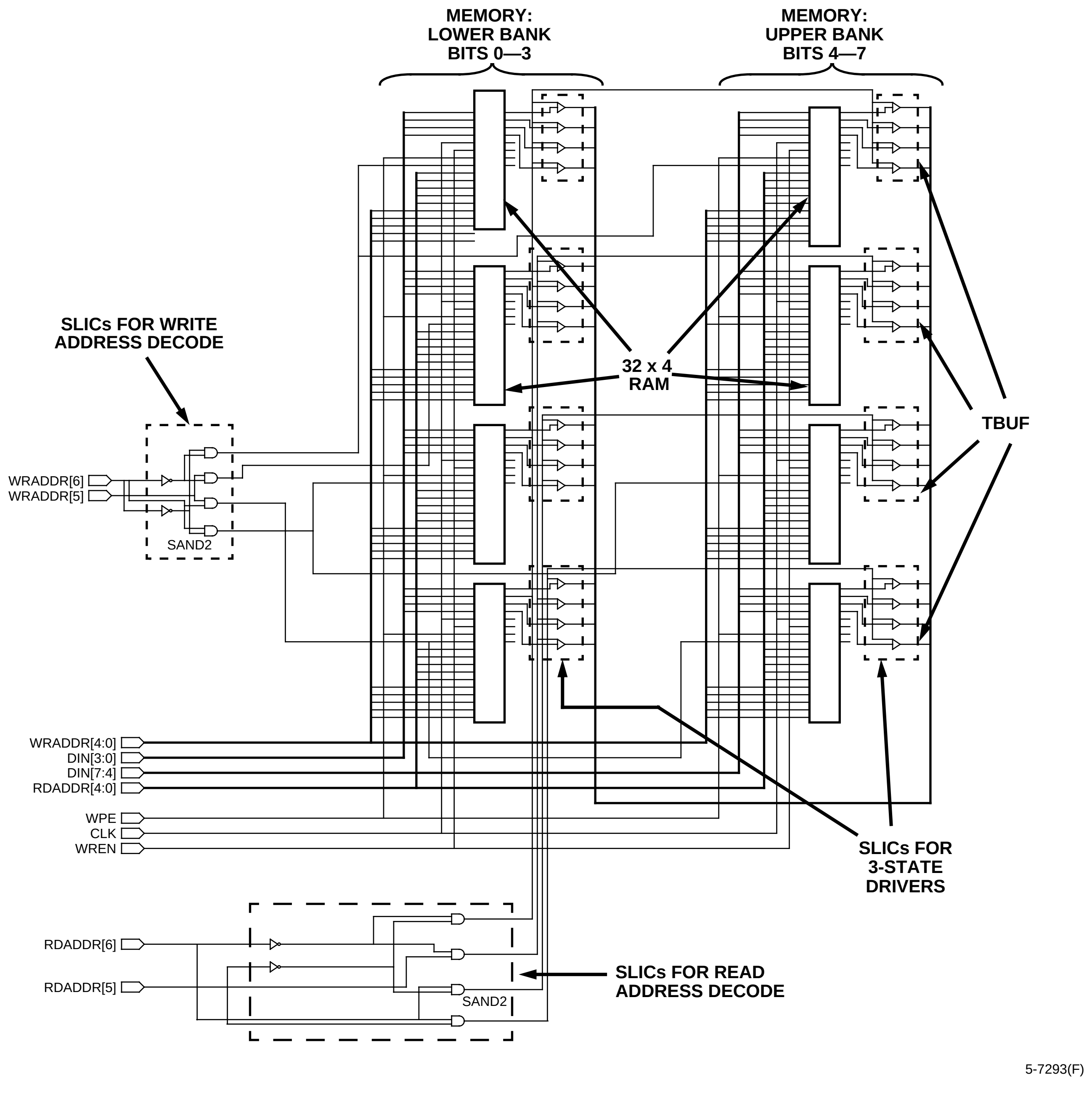
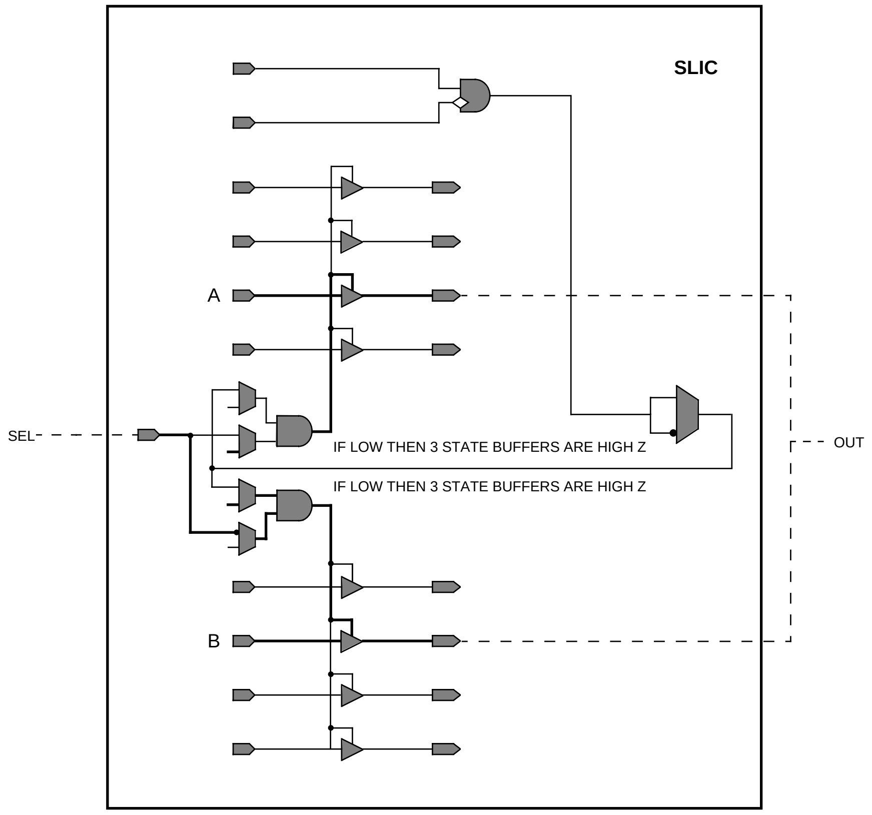

MEMORY: LOWER BANK BITS 0—3
MEMORY: UPPER BANK BITS 4—7
SLICs FOR WRITE ADDRESS DECODE
32 x 4 RAM
TBUF
WRADDR[6]
WRADDR[5]
SAND2
WRADDR[4:0]
DIN[3:0]
DIN[7:4]
RDADDR[4:0]
WPE
CLK
WREN
SLICs FOR 3-STATE DRIVERS
RDADDR[6]
RDADDR[5]
SAND2
SLICs FOR READ ADDRESS DECODE
5-7293(F)

Figure 14. 128 x 8 Dual-Port RAM

Applications (continued)

Implementing Multiplexers with SLIC 3-State Buffers

The SLIC 3-state buffers may also be used to implement very fast multiplexers. Each SLIC can create a 4-bit wide 2 x 1 MUX. They are much faster than LUTs and free up the LUTs for other logic. Figure 15 shows the EPIC drawing of a 1-bit wide 2 x 1 MUX implemented in one SLIC. This may be expanded into a 4-bit wide 4 x 1 MUX using two SLICs or to a 4-bit wide 8 x 1 MUX using four SLICs. Note that the 2-bit decoder shown for group 3 can be used to select and enable the approprate SLIC and/or nibble within a particular SLIC for these larger 4 x 1 and 8 x 1 MUXes.


SLIC
A
SEL
IF LOW THEN 3 STATE BUFFERS ARE HIGH Z
IF LOW THEN 3 STATE BUFFERS ARE HIGH Z
OUT
B

5-7294(F)

Figure 15. 2 x 1 MUX

VHDL Coding of MUXes

The following example illustrates how to implement multiplexers in VHDL using 3-state buffers. The when-else statements infer the 3-state MUX. The else condition of Z triggers selection of the 3-state buffer by the synthesis tool. The code exists outside of the **process** statements. The example code will synthesize 32 3-state buffers.

```
Signal A:        STD_LOGIC_VECTOR (7 downto 0);
Signal B:        STD_LOGIC_VECTOR (7 downto 0);
Signal C:        STD_LOGIC_VECTOR (7 downto 0);
Signal D:        STD_LOGIC_VECTOR (7 downto 0);
Signal OUT:      STD_LOGIC_VECTOR (7 downto 0);
Signal SEL:      STD_LOGIC_VECTOR (1 downto 0);

OUT <= A when (SEL = "00") else (others => ’Z’);
OUT <= B when (SEL = "01") else (others => ’Z’);
OUT <= C when (SEL = "10") else (others => ’Z’);
OUT <= D when (SEL = "11") else (others => ’Z’);
```

Applications (continued)

VHDL SLIC Decode Inference in *Synopsys*†

The following describes a methodology for inferring SLIC decoders from generic VHDL code using a dc shell script in *Synopsys Design Compiler*† or *FPGA Compiler*† without the need for direct instantiation.

Typically, decoding logic and small multiplexers would be implemented using SLIC decoders.

First, logic that must be optimized with SLIC decoders should be isolated. This can be done by either using a new entity (this can be very tedious) or by isolating the logic using the VHDL block statement and then generating a new entity with the following dc shell command:

```
Group—hdl_block B1
```

When using SLIC decoders, the optimization strategy should be the same as the one used for a CPLD/PAL architecture. In *Synopsys,* use the commands:

```
Set_structure false
Set_flatten—effort medium
```

Finally, exclude all non-SLIC gates from the library and include SLIC gates in the library:

```
Set_don’t_use OR3-5.db/AND*
Set_don’t_use OR3-5.db/OR*
Set_don’t_use OR3-5.db/ND*
Set_don’t_use OR3-5.db/NR*

Remove_attribute OR3-5.db/SAND*
Remove_attribute OR3-5.db/SOR*
```

To optimize MUXes with SLIC decoders, the following gates must be added:

```
Remove_attribute OR3-5.db/SAOI*
```

Note: The inverter gate (INV) must not be removed from the library since it is a valid SLIC gate. The *ORCA* Foundry mapper will merge the SLIC decoder gates with inverter gates into the same SLIC block. In addition, to respect *Synopsys* rules, it is required to keep a two-input NOR in the target library.

† *Synopsys* is a registered trademark and *Design Compiler* and *FPGA Compiler* are trademarks of Synopsys Inc.

Applications (continued)

VHDL SLIC Decode Inference in *Synopsys* (continued)

The following example is a counter that drives a 2-bit decoder. The decoder is optimized in a SLIC using a dc shell script from technology independent RTL VHDL code.

```
Libary   ieee;
Use      ieee.std_logic_164.all ;
Use      ieee.std_logic_arith.all
Use      ieee.std_logic_unsigned.all ;

Entity cnt is port (

         Clk       : in std_logic ;
                   : out std_logic_vector(1 downto 0)) ;

End cnt ;

Architecture rlt of cnt is

         Signal count      : std_logic_vector(7 downto 0) ;

Begin
         Process(clk)
         Begin
                   If (clk = '1') and (clk'event) then
                             Count <= count =1 ;
                   End if ;

         End process ;

         DEC1 : block
         Begin
                             Dec(0) <= '1' when (count = "10101010") else '0' ;
                             Dec(1) <= '1' when (count = "01010101") else '0' ;

         End block;

End rlt ;
```

Applications (continued)

VHDL SLIC Decode Inference in *Synopsys* (continued)

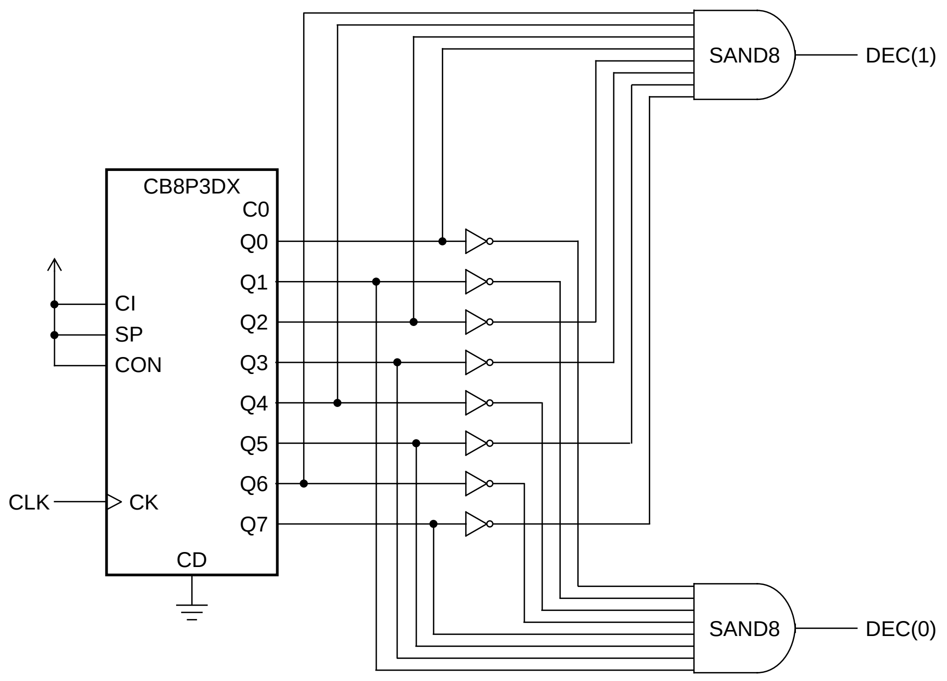
```
/*Logic to be Optimized is Isolated*/
/***********************************/

      group-hgl_block DEC1
      current_design DEC1

/*The target Library is the set to Gate Library for ORCA3*/
/*********************************************************/


      target_library {OR3-5.db}

/*Non SLIC gates are removed from the target library*/
/****************************************************/

      set_dont_use OR3-5.db/AND*
      set_dont_use OR3-5.dn/OR*
      set_dont_use OR3-5.db/NR*
      set_dont_use OR3-5.db/ND*

/*SLIC are inserted in the library*/
/**********************************/

      remove_attribute OR3-5.db/SAND*       don't_use

      remove_attribute OR3-5.db/SOR*        don't_use

/*If a MUX is optimised the Following gates would be inserted*/
/*************************************************************/

      /*remove_attribute OR3-5.db/SAOI*   don't_use*/

/*Logic is flattened*/
/********************/

      set_structure false
      set_flatten true-effort medium

/*Compilation*/
/*************/

      compile - map_effort medium
```

Figure 16. *Synopsys* Script

Applications (continued)

VHDL SLIC Decode Inference in *Synopsys* (continued)

Therefore, it is possible to use dc shell scripts to implement SLIC decoders from technology independent HDL code. This can significantly improve the performance of combinatorial functions and device utilization, thus bypassing the need for manual instantiation.



5-7295(F);

Figure 17. Inferred SLIC Decoder (SAND8)

Summary

The new *ORCA* Series 3 Supplemental Logic and Interconnect Cell (SLIC) has been described in this application note. The primary advantages of speed, density, routability, and flexibility have been highlighted. SLICs provide the ability for fast decode, allow for wide fan-in functions with fewer logic levels than the traditional LUT, and provide 3-state buffering for fast and efficient multiplexing of buses. The SLIC provides flexibility for the designer, allowing for several modes of operation including DECODER, BUFFER, and the ability to mix the modes. Various methods of using the SLIC and some of its applications have been described.

January 2002
AP98-078FPGA

Lattice®
Semiconductor
Corporation